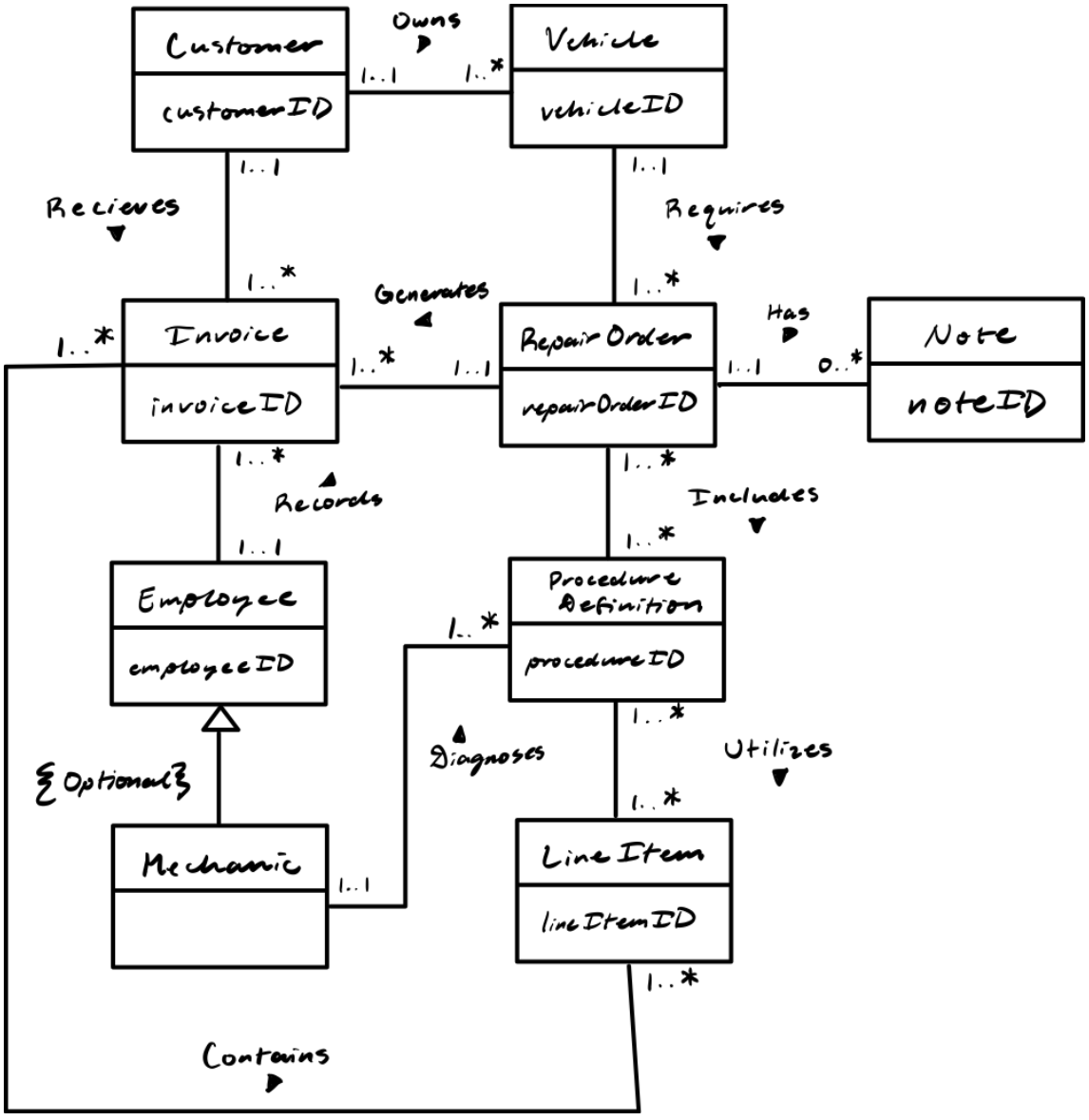
Automotive Repair Information System Report

This project involves designing and implementing a relational database for an automotive repair shop. The system manages customer details, vehicles, repair orders, procedures, invoices, and employees. The goal was to ensure reliable data integrity while enabling comprehensive queries for tracking repair histories, managing billing, and improving the overall efficiency of repair shop operations.

Conceptual Design EER Model

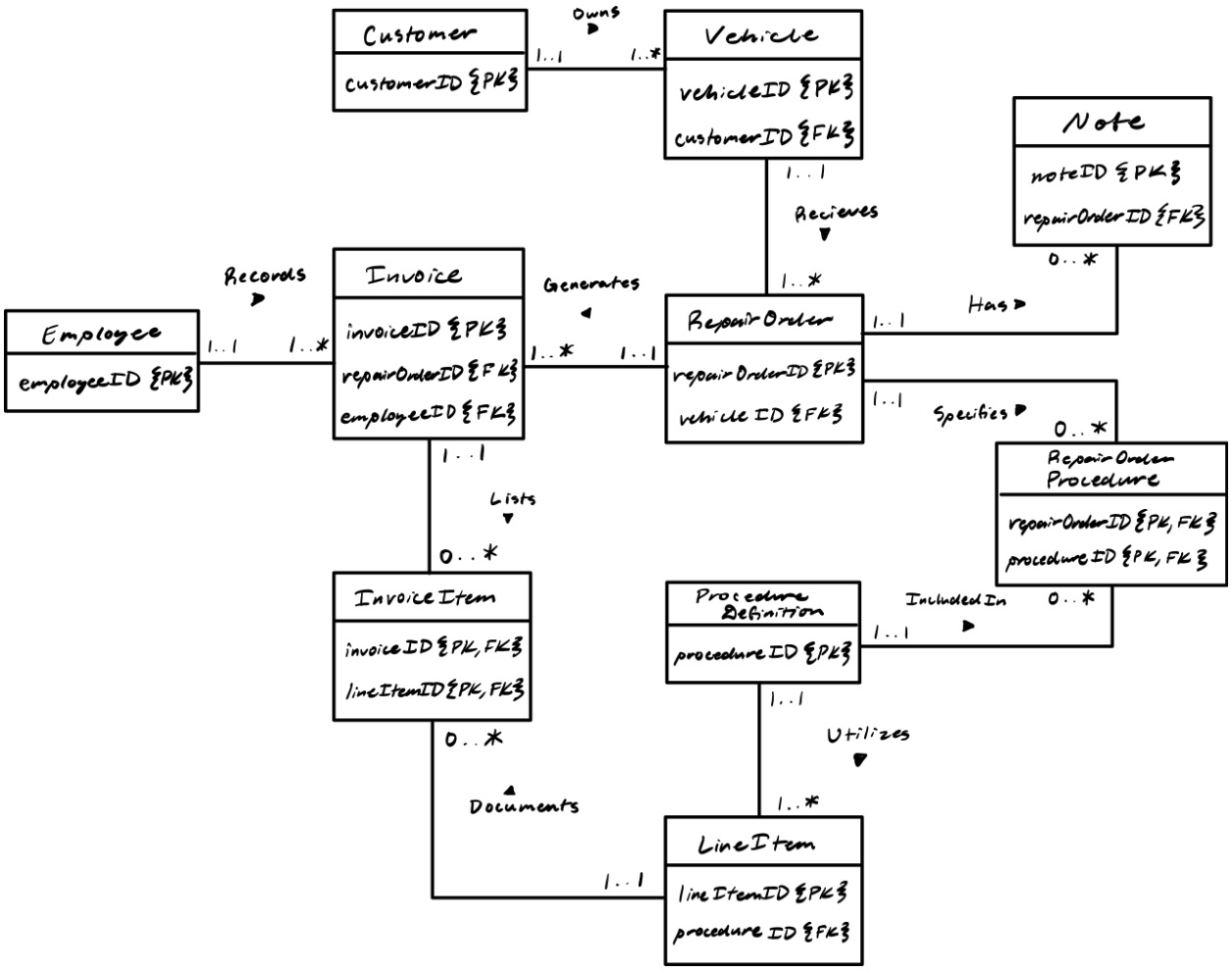
During the conceptual design phase, an Enhanced Entity-Relationship (EER) model was developed. It defines key entities such as Customer, Vehicle, RepairOrder, Invoice, Employee, etc., each with attributes relevant to their roles in the system. The model also illustrates important relationships, such as a vehicle having multiple repair orders, and an invoice containing one or more line items. The conceptual design is represented by the Enhanced Entity-Relationship (EER) model shown below:


Customer
customerID
Owns
1..1
1..*
Vehicle
vehicleID
1..1
Recieves
1..*
1..1
Requires
1..*
Generates
1..*
1..*
Invoice
invoiceID
1..1
Repair Order
repairOrderID
Has
1..1
0..*
Note
noteID
1..*
1..*
Records
1..1
1..*
Includes
1..*
Employee
employeeID
1..*
Procedure Definition
procedureID
{Optional}
Diagnoses
1..*
Utilizes
1..*
Mechanic
1..1
Line Item
lineItemID
1..*
Contains

Logical Design

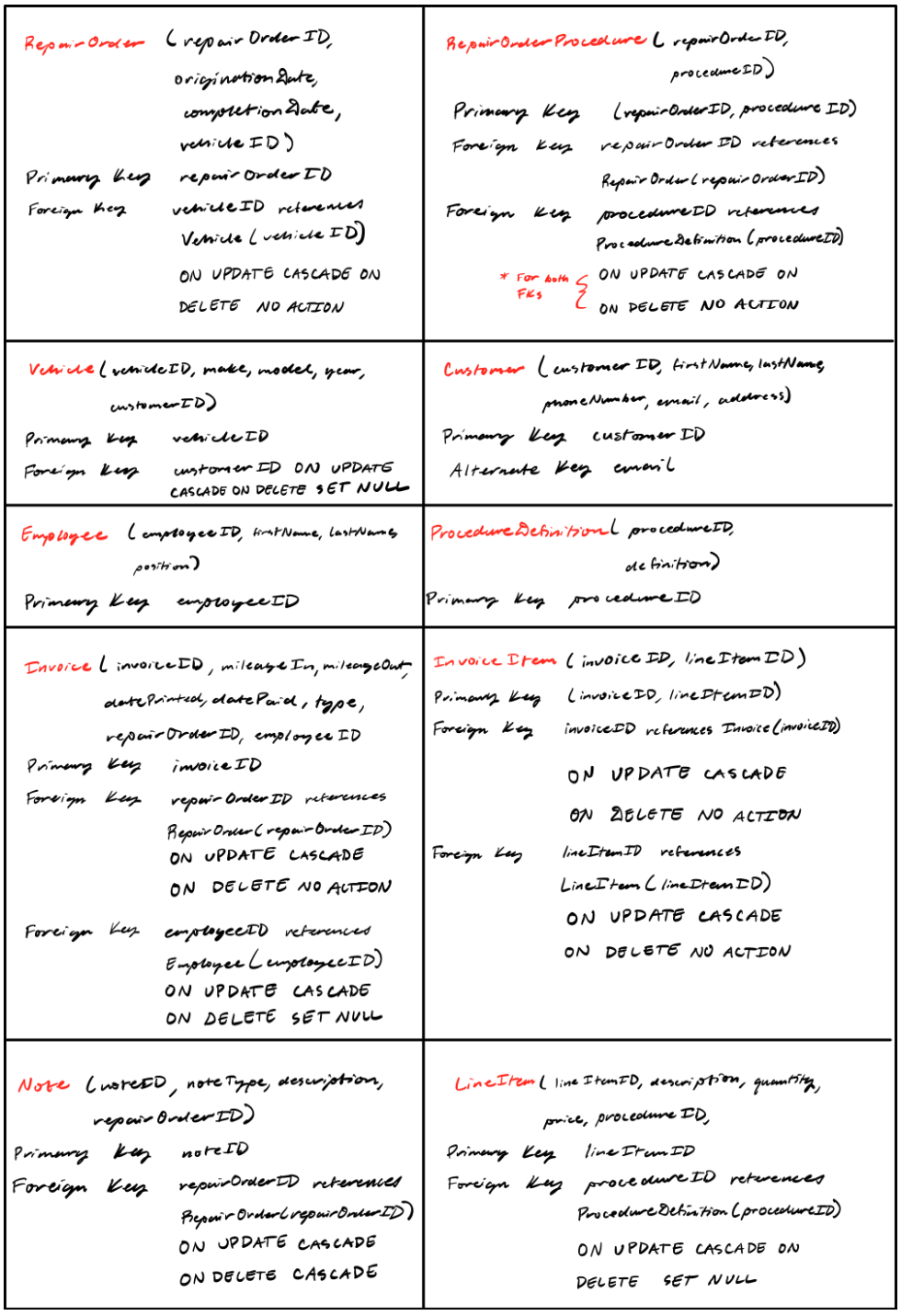
EER Model

Similarly, the logical model represents the relational schema derived from the conceptual EER diagram, now enhanced with foreign keys and further normalized to ensure it is in Boyce-Codd Normal Form (BCNF). The logical EER model is shown below:



Data Model

Additionally, the logical data model, shown below, includes all relevant entities, their attributes, and the relationships necessary to support repair tracking, customer management, and invoicing within the vehicle repair shop.



SQL Implementation

To implement the logical data model, the following SQL statements were used to define the database schema in PostgreSQL, creating the necessary tables and relationships to manage customers, vehicles, repair orders, invoices, and employees.

Customer Table

SQL Statements for Creation and Insertion

```
CREATE TABLE Customer (
    customerID SERIAL PRIMARY KEY,
    firstName VARCHAR (100) NOT NULL,
    lastName VARCHAR (100) NOT NULL,
    phoneNumber CHAR(10),
    email VARCHAR (100) UNIQUE,
    address VARCHAR (200)
);

INSERT INTO Customer (firstName, lastName, phoneNumber, email, address) VALUES
('Elena', 'Clark', '4183091439', 'eclark@email.com', '132 Maplewood Dr, Springfield, IL 62704'),
('Chloe', 'Taylor', '1585384951', 'ctaylor@email.com', '87 Crestview Ln, Austin, TX 73301'),
('Lucy', 'Harris', '9135531857', 'lharris@email.com', '2457 Riverbend Rd, Columbus, OH 43215'),
('Alexander', 'Allen', '9475195848', 'aallen@email.com', '612 Winding Way, Denver, CO 80203'),
('Carter', 'Scott', '9401841240', 'cscott@email.com', '498 Hilltop Ct, Seattle, WA 98101'),
('James', 'Parker', '9280940198', 'jparker@email.com', '9307 Birchwood Ave, Orlando, FL 32801'),
('Aiden', 'Cook', '2092097298', 'acook@email.com', '1128 Aspen Trail, Phoenix, AZ 85001'),
('Aurora', 'Kelly', '7810528417', 'akelly@email.com', '3360 Oak Knoll Dr, Atlanta, GA 30303');
```

Test Data for Customer

```
SELECT * FROM Customer;
```

	customerid [PK] integer	firstname character varying (100)	lastname character varying (100)	phonenumber character (10)	email character varying (100)	address character varying (200)
1	1	Elena	Clark	4183091439	eclark@email.com	132 Maplewood Dr, Springfield, IL 62704
2	2	Chloe	Taylor	1585384951	ctaylor@email.com	87 Crestview Ln, Austin, TX 73301
3	3	Lucy	Harris	9135531857	lharris@email.com	2457 Riverbend Rd, Columbus, OH 432...
4	4	Alexander	Allen	9475195848	aallen@email.com	612 Winding Way, Denver, CO 80203
5	5	Carter	Scott	9401841240	cscott@email.com	498 Hilltop Ct, Seattle, WA 98101
6	6	James	Parker	9280940198	jparker@email.com	9307 Birchwood Ave, Orlando, FL 32801
7	7	Aiden	Cook	2092097298	acook@email.com	1128 Aspen Trail, Phoenix, AZ 85001
8	8	Aurora	Kelly	7810528417	akelly@email.com	3360 Oak Knoll Dr, Atlanta, GA 30303

Vehicle Table

SQL Statements for Creation and Insertion

```sql
CREATE TABLE Vehicle (
    vehicleID SERIAL PRIMARY KEY,
    make VARCHAR(100) NOT NULL,
    model VARCHAR(100) NOT NULL,
    year INT NOT NULL,
    customerID INT NOT NULL,
    FOREIGN KEY (customerID) REFERENCES Customer(customerID)
    ON UPDATE CASCADE
    ON DELETE SET NULL
);

INSERT INTO Vehicle (make, model, year, customerID) VALUES
('Toyota', 'Camry', '2020', 1),
('Toyota', 'Corolla', '2019', 2),
('Honda', 'Civic', '2021', 3),
('Honda', 'Accord', '2022', 4),
('Nissan', 'Altima', '2018', 5),
('Nissan', 'Rogue', '2023', 6),
('BMW', '3 Series', '2022', 7),
('BMW', 'X5', '2021', 8),
('Mazda', 'CX-5', '2022', 1),
('Mazda', 'Mazda3', '2021', 2);
```

Test Data for Vehicle

```sql
SELECT * FROM Vehicle;
```

	vehicleid [PK] integer	make character varying (100)	model character varying (100)	year integer	customerid integer
1	1	Toyota	Camry	2020	1
2	2	Toyota	Corolla	2019	2
3	3	Honda	Civic	2021	3
4	4	Honda	Accord	2022	4
5	5	Nissan	Altima	2018	5
6	6	Nissan	Rogue	2023	6
7	7	BMW	3 Series	2022	7
8	8	BMW	X5	2021	8
9	9	Mazda	CX-5	2022	1
10	10	Mazda	Mazda3	2021	2

ProcedureDefinition Table

SQL Statements for Creation and Insertion

```
CREATE TABLE ProcedureDefinition (
    procedureID SERIAL PRIMARY KEY,
    definition VARCHAR(100) NOT NULL
);

INSERT INTO ProcedureDefinition(definition) VALUES
('Oil Change'),
('Brake Pad Replacement'),
('Tire Rotation and Balance'),
('Battery Replacement'),
('Air Conditioning Service'),
('Engine Tune-Up'),
('Transmission Fluid Flush'),
('Wheel Alignment'),
('Alternator Replacement'),
('Timing Belt Replacement');
```

Test Data for ProcedureDefinition

```
SELECT * FROM ProcedureDefinition;
```

	procedureid [PK] integer	definition character varying (100)
1	1	Oil Change
2	2	Brake Pad Replacement
3	3	Tire Rotation and Balance
4	4	Battery Replacement
5	5	Air Conditioning Service
6	6	Engine Tune-Up
7	7	Transmission Fluid Flush
8	8	Wheel Alignment
9	9	Alternator Replacement
10	10	Timing Belt Replacement

RepairOrder Table

SQL Statements for Creation and Insertion

```
CREATE TABLE RepairOrder(
    repairOrderID SERIAL PRIMARY KEY,
    originationDate DATE NOT NULL,
    completionDate DATE,
    vehicleID INT NOT NULL,
    FOREIGN KEY (vehicleID) REFERENCES Vehicle(vehicleID)
    ON UPDATE CASCADE
    ON DELETE NO ACTION
);

INSERT INTO RepairOrder (originationDate, completionDate, vehicleID) VALUES
('2024-01-12', '2024-01-14', 1),
('2024-03-05', '2024-03-10', 2),
('2024-05-18', '2024-05-20', 3),
('2024-07-09', '2024-07-14', 4),
('2024-08-22', '2024-08-23', 5),
('2024-09-10', '2024-09-13', 6),
('2024-10-03', '2024-10-07', 7),
('2024-11-11', '2024-11-12', 8),
('2024-12-01', '2024-12-04', 9),
('2024-12-20', '2024-12-22', 10),
('2024-07-18', '2024-07-20', 1),
('2024-09-13', '2024-09-18', 1),
('2024-12-01', '2024-12-04', 9);
```

Test Data for RepairOrder

```
SELECT * FROM RepairOrder;
```

	repairorderid [PK] integer	originationdate date	completiondate date	vehicleid integer
1	1	2024-01-12	2024-01-14	1
2	2	2024-03-05	2024-03-10	2
3	3	2024-05-18	2024-05-20	3
4	4	2024-07-09	2024-07-14	4
5	5	2024-08-22	2024-08-23	5
6	6	2024-09-10	2024-09-13	6
7	7	2024-10-03	2024-10-07	7
8	8	2024-11-11	2024-11-12	8
9	9	2024-12-01	2024-12-04	9
10	10	2024-12-20	2024-12-22	10
11	11	2024-07-18	2024-07-20	1
12	12	2024-09-13	2024-09-18	1
13	13	2024-12-01	2024-12-04	9

Note Table

SQL Statements for Creation and Insertion

```
CREATE TABLE Note(
    noteID SERIAL PRIMARY KEY,
    noteType VARCHAR(100) NOT NULL,
    description VARCHAR(100) NOT NULL,
    repairOrderID INT NOT NULL,
    FOREIGN KEY (repairOrderID) REFERENCES RepairOrder(repairOrderID)
    ON UPDATE CASCADE
    ON DELETE CASCADE
);

INSERT INTO Note (noteType, description, repairOrderID) VALUES
('complaint', 'Customer reports engine takes several tries to start in the morning, especially in cold weather', 1),
('complaint', 'Brakes are squealing loudly', 2),
('documentation', 'Diagnosed weak battery and corroded terminals.', 1),
('complaint', 'Air conditioning not blowing cold air; fan works but no cool air is coming out.', 3),
('complaint', 'Battery dies overnight unless disconnected', 4);
```

Test Data for Note

```
SELECT * FROM Note;
```

	noteid [PK] integer	notetype character varying (100)	description character varying (100)	repairorderid integer
1	1	complaint	Customer reports engine takes several tries to start in the morning, especially in cold weather	1
2	2	complaint	Brakes are squealing loudly	2
3	3	documentation	Diagnosed weak battery and corroded terminals.	1
4	4	complaint	Air conditioning not blowing cold air; fan works but no cool air is coming out.	3
5	5	complaint	Battery dies overnight unless disconnected	4

Employee Table

SQL Statements for Creation and Insertion

```
CREATE TABLE Employee(
    employeeID SERIAL PRIMARY KEY,
    firstName VARCHAR(100) NOT NULL,
    lastName VARCHAR (100) NOT NULL,
    position VARCHAR(100)
);

INSERT INTO Employee (firstName, lastName, position)
('Henry', 'Brown', 'Mechanic'),
('William', 'Thompson', 'Mechanic'),
('Owen', 'Lee', NULL),
('Lucy', 'Young', NULL),
('Miles', 'Cooper', 'Mechanic'),
('Erza', 'Cox', 'Mechanic');
```

Test Data for Employee

```
SELECT * FROM Employee;
```

	employeeid [PK] integer	firstname character varying (100)	lastname character varying (100)	position character varying (100)
1	1	Henry	Brown	Mechanic
2	2	William	Thompson	Mechanic
3	3	Owen	Lee	[null]
4	4	Lucy	Young	[null]
5	5	Miles	Cooper	Mechanic
6	6	Erza	Cox	Mechanic

RepairOrderProcedure Table

SQL Statements for Creation and Insertion

```
CREATE TABLE RepairOrderProcedure(
    repairOrderID INT NOT NULL,
    procedureID INT NOT NULL,
    PRIMARY KEY (repairOrderID, procedureID),
    FOREIGN KEY (repairOrderID) REFERENCES RepairOrder(repairOrderID)
    ON UPDATE CASCADE
    ON DELETE NO ACTION,
    FOREIGN KEY (procedureID) REFERENCES ProcedureDefinition(procedureID)
    ON UPDATE CASCADE
    ON DELETE NO ACTION
);

INSERT INTO RepairOrderProcedure(repairOrderID, procedureID) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),
(6, 6),
(7, 7),
(8, 8),
(9, 9),
(10, 10),
(1, 2),
(2, 5);
```

Test Data for RepairOrderProcedure

```
SELECT * FROM RepairOrderProcedure;
```

	repairorderid [PK] integer	procedureid [PK] integer
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10
11	1	2
12	2	5

Invoice Table

SQL Statements for Creation and Insertion

```sql
-- Create Invoice Table
CREATE TABLE Invoice (
    invoiceID SERIAL PRIMARY KEY,
    mileageIn INT NOT NULL,
    mileageOut INT NOT NULL,
    datePrinted DATE,
    datePaid DATE,
    type VARCHAR (100),
    repairOrderID INT NOT NULL,
    employeeID INT,
    FOREIGN KEY (repairOrderID) REFERENCES RepairOrder(repairOrderID)
    ON UPDATE CASCADE
    ON DELETE NO ACTION,
    FOREIGN KEY (employeeID) REFERENCES Employee(employeeID)
    ON UPDATE CASCADE
    ON DELETE SET NULL
);
```

```sql
-- Insert invoices
INSERT INTO Invoice (mileageIn, mileageOut, datePrinted, datePaid, type, repairOrderID, employeeID) VALUES
(87450, 87455, '2024-01-14', '2024-01-01', 'Final Bill', 1, 1),
(104320, 104325, '2024-03-10', '2024-03-11', 'Partial Billing', 2, 2),
(61780, 61784, '2024-05-20', '2024-05-21', 'Final Bill', 3, 3),
(129990, 129996, '2024-07-14', '2024-07-15', 'Prepayment', 4, 4),
(45610, 45615, '2024-08-23', '2024-08-23', 'Final Bill', 5, 5),
(78234, 78239, '2024-09-13', '2024-09-15', 'Partial Billing', 6, 6),
(152110, 152118, '2024-10-07', '2024-10-09', 'Final Bill', 7, 6),
(91003, 91009, '2024-11-12', '2024-11-12', 'Prepayment', 8, 1),
(110800, 110807, '2024-12-04', '2024-12-04', 'Final Bill', 9, 4),
(69750, 69755, '2024-12-22', '2024-12-23', 'Partial Billing', 10, 6),
(78234, 78239, '2024-09-20', '2024-09-30', 'Final Billing', 6, 6);
```

Test Data for Invoice

```sql
-- View invoices
SELECT * FROM Invoice;
```

	invoiceid [PK] integ	mileagein integer	mileageout integer	dateprinted date	datepaid date	type character varying	repairor integer	employe integer
1	1	87450	87455	2024-01-14	2024-01-01	Final Bill	1	1
2	2	104320	104325	2024-03-10	2024-03-11	Partial Billing	2	2
3	3	61780	61784	2024-05-20	2024-05-21	Final Bill	3	3
4	4	129990	129996	2024-07-14	2024-07-15	Prepayment	4	4
5	5	45610	45615	2024-08-23	2024-08-23	Final Bill	5	5
6	6	78234	78239	2024-09-13	2024-09-15	Partial Billing	6	6
7	7	152110	152118	2024-10-07	2024-10-09	Final Bill	7	6
8	8	91003	91009	2024-11-12	2024-11-12	Prepayment	8	1
9	9	110800	110807	2024-12-04	2024-12-04	Final Bill	9	4
10	10	69750	69755	2024-12-22	2024-12-23	Partial Billing	10	6
11	11	78234	78239	2024-09-20	2024-09-30	Final Billing	6	6

LineItem Table

SQL Statements for Creation and Insertion

```
CREATE TABLE LineItem (
    lineItemID SERIAL PRIMARY KEY,
    description VARCHAR(100),
    quantity INT,
    price DECIMAL (10, 2),
    procedureID INT NOT NULL,
    FOREIGN KEY (procedureID) REFERENCES ProcedureDefinition(procedureID)
    ON UPDATE CASCADE
    ON DELETE SET NULL
);
```

```
INSERT INTO LineItem (description, quantity, price, procedureID) VALUES
('Labor for oil change', 1, 30, 1),
('Engine oil', 5, 6, 1),
('Oil filter', 1, 12, 1),
('Labor for brake pad replacement', 1, 80, 2),
('Front brake pads', 1, 45, 2),
('Brake cleaner', 1, 5, 2),
('Labor for tire rotation', 1, 25, 3),
('Tire balancing', 4, 10, 3),
('Battery', 1, 120, 4),
('Labor for battery replacement', 1, 20, 4),
('Labor for A/C inspection', 1, 60, 5),
('Refrigerant', 2, 15, 5),
('A/C system recharge and test', 1, 30, 5),
('Spark plugs', 4, 10, 6),
('Labor for tune-up', 1, 100, 6),
('Fuel system cleaner', 1, 15, 6),
('Labor for fluid flush', 1, 90, 7),
('Transmission fluid', 6, 9, 7),
('Labor for alignment', 1, 85, 8),
('Alignment inspection', 1, 20, 8),
('Alternator', 1, 180, 9),
('Labor for replacement', 1, 90, 9),
('Drive belt', 1, 20, 9),
('Timing belt kit', 1, 150, 10),
('Coolant', 1, 15, 10);
```

Test Data for LineItem

```
SELECT * FROM LineItem;
```

	lineitemid [PK] integer	description character varying (100)	quantity integer	price numeric (10,2)	procedureid integer
1	1	Labor for oil change	1	30.00	1
2	2	Engine oil	5	6.00	1
3	3	Oil filter	1	12.00	1
4	4	Labor for brake pad replacement	1	80.00	2
5	5	Front brake pads	1	45.00	2
6	6	Brake cleaner	1	5.00	2
7	7	Labor for tire rotation	1	25.00	3
8	8	Tire balancing	4	10.00	3
9	9	Battery	1	120.00	4
10	10	Labor for battery replacement	1	20.00	4
11	11	Labor for A/C inspection	1	60.00	5
12	12	Refrigerant	2	15.00	5
13	13	A/C system recharge and test	1	30.00	5
14	14	Spark plugs	4	10.00	6
15	15	Labor for tune-up	1	100.00	6

```
SELECT * FROM LineItem;
```

16	16	Fuel system cleaner	1	15.00	6
17	17	Labor for fluid flush	1	90.00	7
18	18	Transmission fluid	6	9.00	7
19	19	Labor for alignment	1	85.00	8
20	20	Alignment inspection	1	20.00	8
21	21	Alternator	1	180.00	9
22	22	Labor for replacement	1	90.00	9
23	23	Drive belt	1	20.00	9
24	24	Timing belt kit	1	150.00	10
25	25	Labor for replacement	1	250.00	10
26	26	Coolant	1	15.00	10

InvoiceItem Table

SQL Statements for Creation and Insertion

```sql
CREATE TABLE InvoiceItem (
    invoiceID INT NOT NULL,
    lineItemID INT NOT NULL,
    PRIMARY KEY (invoiceID, lineItemID),
    FOREIGN KEY (invoiceID) REFERENCES Invoice(invoiceID)
    ON UPDATE CASCADE
    ON DELETE NO ACTION,
    FOREIGN KEY (lineItemID) REFERENCES LineItem(lineItemID)
    ON UPDATE CASCADE
    ON DELETE NO ACTION
);

INSERT INTO InvoiceItem (invoiceID, lineItemID) VALUES
-- Invoice 1: Oil Change + Brake Pad Replacement
(1, 1), (1, 2), (1, 3),  -- Oil Change
(1, 4), (1, 5), (1, 6),  -- Brake Pad Replacement

-- Invoice 2: Brake Pad Replacement + Air Conditioning Service
(2, 4), (2, 5), (2, 6),  -- Brake Pad Replacement
(2, 11), (2, 12), (2, 13),  -- A/C Service

-- Invoice 3: Tire Rotation and Balance
(3, 7), (3, 8),

-- Invoice 4: Battery Replacement
(4, 9), (4, 10),

-- Invoice 5: Air Conditioning Service
(5, 11), (5, 12), (5, 13),

-- Invoice 6: Engine Tune-Up
(6, 14), (6, 15), (6, 16),

-- Invoice 7: Transmission Fluid Flush
(7, 17), (7, 18),


-- Invoice 9: Alternator Replacement
(9, 21), (9, 22), (9, 23),

-- Invoice 10: Timing Belt Replacement
(10, 24), (10, 25), (10, 26),

-- Invoice 11: Wheel Alignment
(11, 19), (11, 20);
```

Test Data for InvoiceItem

```
SELECT * FROM InvoiceItem;
```

	invoiceid [PK] integer	lineitemid [PK] integer
1	1	1
2	1	2
3	1	3
4	1	4
5	1	5
6	1	6
7	2	4
8	2	5
9	2	6
10	2	11
11	2	12
12	2	13
13	3	7
14	3	8
15	4	9

```
SELECT * FROM InvoiceItem;
```

16	4	10
17	5	11
18	5	12
19	5	13
20	6	14
21	6	15
22	6	16
23	7	17
24	7	18
25	8	19
26	8	20
27	9	21
28	9	22
29	9	23
30	10	24

```
-- View invoice items
SELECT * FROM InvoiceItem;
```

31	10	25
32	10	26
33	11	19
34	11	20

SQL Statements and Output for Sample Queries

The following section presents the sample SQL queries, along with screenshots that demonstrate the results of each query execution.

1. Insert a new customer into the database

   Customer table before insertion:

```
SELECT * FROM Customer;
```

	customerid [PK] integer	firstname character varying (100)	lastname character varying (100)	phonenumber character (10)	email character varying (100)	address character varying (200)
1	1	Elena	Clark	4183091439	eclark@email.com	132 Maplewood Dr, Springfield, IL 62704
2	2	Chloe	Taylor	1585384951	ctaylor@email.com	87 Crestview Ln, Austin, TX 73301
3	3	Lucy	Harris	9135531857	lharris@email.com	2457 Riverbend Rd, Columbus, OH 432...
4	4	Alexander	Allen	9475195848	aallen@email.com	612 Winding Way, Denver, CO 80203
5	5	Carter	Scott	9401841240	cscott@email.com	498 Hilltop Ct, Seattle, WA 98101
6	6	James	Parker	9280940198	jparker@email.com	9307 Birchwood Ave, Orlando, FL 32801
7	7	Aiden	Cook	2092097298	acook@email.com	1128 Aspen Trail, Phoenix, AZ 85001
8	8	Aurora	Kelly	7810528417	akelly@email.com	3360 Oak Knoll Dr, Atlanta, GA 30303

   Customer Table after insertion:

```
-- Sample Query 1: Adding new customer
INSERT INTO Customer (firstName, lastName, phoneNumber, email, address) VALUES
('Joe', 'Quang', '8138491239', 'jquang@email.com', '742 Evergreen Terrance, Springfield, IL 62704');
```

```
SELECT * FROM Customer;
```

	customerid [PK] integer	firstname character varying (100)	lastname character varying (100)	phonenumber character (10)	email character varying (100)	address character varying (200)
1	1	Elena	Clark	4183091439	eclark@email.com	132 Maplewood Dr, Springfield, IL 62704
2	2	Chloe	Taylor	1585384951	ctaylor@email.com	87 Crestview Ln, Austin, TX 73301
3	3	Lucy	Harris	9135531857	lharris@email.com	2457 Riverbend Rd, Columbus, OH 43215
4	4	Alexander	Allen	9475195848	aallen@email.com	612 Winding Way, Denver, CO 80203
5	5	Carter	Scott	9401841240	cscott@email.com	498 Hilltop Ct, Seattle, WA 98101
6	6	James	Parker	9280940198	jparker@email.com	9307 Birchwood Ave, Orlando, FL 32801
7	7	Aiden	Cook	2092097298	acook@email.com	1128 Aspen Trail, Phoenix, AZ 85001
8	8	Aurora	Kelly	7810528417	akelly@email.com	3360 Oak Knoll Dr, Atlanta, GA 30303
9	9	Joe	Quang	8138491239	jquang@email.com	742 Evergreen Terrance, Springfield, IL 62704

2. Delete an existing employee from the database

   Employee table before deletion:

```
SELECT * FROM Employee;
```

	employeeid [PK] integer	firstname character varying (100)	lastname character varying (100)	position character varying (100)
1	1	Henry	Brown	Mechanic
2	2	William	Thompson	Mechanic
3	3	Owen	Lee	[null]
4	4	Lucy	Young	[null]
5	5	Miles	Cooper	Mechanic
6	6	Erza	Cox	Mechanic

Employee table after deletion:

```
-- Sample Query 2: Deleting employee 3
DELETE FROM Employee
WHERE employeeID = 3;
```

	employeeid [PK] integer	firstname character varying (100)	lastname character varying (100)	position character varying (100)
1	1	Henry	Brown	Mechanic
2	2	William	Thompson	Mechanic
3	4	Lucy	Young	[null]
4	5	Miles	Cooper	Mechanic
5	6	Erza	Cox	Mechanic

3. Update the description of an existing repair order

   ** Description of repair orders is kept in the notes

   Note table before update:

```
SELECT * FROM Note;
```

	noteid [PK] integer	notetype character varying (100)	description character varying (100)	repairorderid integer
1	1	complaint	Customer reports engine takes several tries to start in the morning, especially in cold weather	1
2	2	complaint	Brakes are squealing loudly	2
3	3	documentation	Diagnosed weak battery and corroded terminals.	1
4	4	complaint	Air conditioning not blowing cold air; fan works but no cool air is coming out.	3
5	5	complaint	Battery dies overnight unless disconnected	4

   Note table after update:

```
-- Sample Query 3: Given repairOrderID 1
UPDATE Note
SET description = 'Engine Oil needs to be topped up.'
WHERE repairOrderID = 1 and noteID = 1;
```

	noteid [PK] intege	notetype character varying (	description character varying (100)	repairorderid integer
1	2	complaint	Brakes are squealing loudly	2
2	3	documentation	Diagnosed weak battery and corroded terminals.	1
3	4	complaint	Air conditioning not blowing cold air; fan works but no cool air is coming out.	3
4	5	complaint	Battery dies overnight unless disconnected	4
5	1	complaint	Engine Oil needs to be topped up.	1

4. List all the repair orders belonging to a given vehicle, along with their dates when each repair was originated and completed.

```
-- Query 4: Given vehicle 1
SELECT repairOrderID, originationDate, completionDate
FROM RepairOrder
WHERE vehicleID = 1;
```

	repairorderid [PK] integer	originationdate date	completiondate date
1	1	2024-01-12	2024-01-14
2	11	2024-07-18	2024-07-20
3	12	2024-09-13	2024-09-18

```
-- Query 4: Given vehicle 5
SELECT repairOrderID, originationDate, completionDate
FROM RepairOrder
WHERE vehicleID = 5;
```

	repairorderid [PK] integer	originationdate date	completiondate date
1	5	2024-08-22	2024-08-23

5. List the details of the line items including the description, price, and quantity for the invoice(s) of a given repair order.

```
-- Sample Query 5: Given repair order 6
-- mulitple invoices
SELECT li.description, li.price, li.quantity
FROM LineItem li
JOIN InvoiceItem ii ON li.lineItemID = ii.lineItemID
JOIN Invoice i ON ii.invoiceID = i.invoiceID
WHERE i.repairOrderID = 6;
```

	description character varying (100)	price numeric (10,2)	quantity integer
1	Spark plugs	10.00	4
2	Labor for tune-up	100.00	1
3	Fuel system cleaner	15.00	1
4	Labor for alignment	85.00	1
5	Alignment inspection	20.00	1

```
-- Sample Query 5: Given repair order 1
-- one invoice
SELECT li.description, li.price, li.quantity
FROM LineItem li
JOIN InvoiceItem ii ON li.lineItemID = ii.lineItemID
JOIN Invoice i ON ii.invoiceID = i.invoiceID
WHERE i.repairOrderID = 1;
```

	description character varying (100)	price numeric (10,2)	quantity integer
1	Labor for oil change	30.00	1
2	Engine oil	6.00	5
3	Oil filter	12.00	1
4	Labor for brake pad replacement	80.00	1
5	Front brake pads	45.00	1
6	Brake cleaner	5.00	1

6. List the repair orders completed between June 2024 and December 2024, sorted by the repair order numbers.

```
-- Sample Query 6: Sorted by ASC
SELECT repairOrderID, completionDate
FROM RepairOrder
WHERE completionDate BETWEEN '2024-06-01' AND '2024-12-31'
ORDER BY repairOrderID ASC;
```

	repairorderid [PK] integer	completiondate date
1	4	2024-07-14
2	5	2024-08-23
3	6	2024-09-13
4	7	2024-10-07
5	8	2024-11-12
6	9	2024-12-04
7	10	2024-12-22
8	11	2024-07-20
9	12	2024-09-18
10	13	2024-12-04

```
-- Sample Query 6: Sorted by DESC
SELECT repairOrderID, completionDate
FROM RepairOrder
WHERE completionDate BETWEEN '2024-06-01' AND '2024-12-31'
ORDER BY repairOrderID DESC;
```

	repairorderid [PK] integer	completiondate date
1	13	2024-12-04
2	12	2024-09-18
3	11	2024-07-20
4	10	2024-12-22
5	9	2024-12-04
6	8	2024-11-12
7	7	2024-10-07
8	6	2024-09-13
9	5	2024-08-23
10	4	2024-07-14

7. List the details of all the line items of a given procedure

```
-- Sample Query 7: Given procedure 1
SELECT description, quantity, price
FROM LineItem
WHERE procedureID = 1;
```

	description character varying (100)	quantity integer	price numeric (10,2)
1	Labor for oil change	1	30.00
2	Engine oil	5	6.00
3	Oil filter	1	12.00

```
-- Sample Query 7: Given procedure 3
SELECT description, quantity, price
FROM LineItem
WHERE procedureID = 3;
```

	description character varying (100)	quantity integer	price numeric (10,2)
1	Labor for tire rotation	1	25.00
2	Tire balancing	4	10.00

8. List the total number of procedures required by each repair order in descending order, along with the procedure description.

```
-- Sample Query 8:
SELECT ro.repairOrderID,
       STRING_AGG(pd.definition, ', ')
       AS procedureDescriptions,
       COUNT(rop.procedureID) AS procedureCount
FROM RepairOrderProcedure rop
JOIN ProcedureDefinition pd ON rop.procedureID = pd.procedureID
JOIN RepairOrder ro ON ro.repairOrderID = rop.repairOrderID
GROUP BY ro.repairOrderID
ORDER BY procedureCount DESC;
```

	repairorderid [PK] integer	proceduredescriptions text	procedurecount bigint
1	1	Oil Change, Brake Pad Replacement	2
2	2	Air Conditioning Service, Brake Pad Replacement	2
3	3	Tire Rotation and Balance	1
4	4	Battery Replacement	1
5	5	Air Conditioning Service	1
6	6	Engine Tune-Up	1
7	7	Transmission Fluid Flush	1
8	8	Wheel Alignment	1
9	9	Alternator Replacement	1
10	10	Timing Belt Replacement	1

9. List the name of the employee who recorded more than the average number of invoices, together with the number of invoices he/she recorded.

```
-- Sample Query 9:
SELECT e.firstName, e.lastName, COUNT(i.invoiceID) AS invoiceCount
FROM Employee e
JOIN Invoice i ON e.employeeID = i.employeeID
GROUP BY e.employeeID
HAVING COUNT(i.invoiceID) > (
    SELECT AVG(invoiceCount)
    FROM (
        SELECT COUNT(invoiceID) AS invoiceCount
        FROM Invoice
        GROUP BY employeeID
    ) sub
);
```

	firstname character varying (100)	lastname character varying (100)	invoicecount bigint
1	Lucy	Young	2
2	Erza	Cox	4
3	Henry	Brown	2

10. For a particular invoice, list the odometer mileages (in and out), payment information, and the vehicle information.

```
-- Sample Query 10: Given invoice 1
SELECT i.mileageIn, i.mileageOut, i.datePrinted, i.datePaid, i.type,
       v.vehicleID, v.make, v.model, v.year
FROM Invoice i
JOIN RepairOrder ro ON i.repairOrderID = ro.repairOrderID
JOIN Vehicle v ON ro.vehicleID = v.vehicleID
WHERE i.invoiceID = 1;
```

	mileagein integer	mileageout integer	dateprinted date	datepaid date	type character varying	vehicleid integer	make character varyir	model character va	year integer
1	87450	87455	2024-01-14	2024-01-01	Final Bill	1	Toyota	Camry	2020

```
-- Sample Query 10: Given invoice 4
SELECT i.mileageIn, i.mileageOut, i.datePrinted, i.datePaid, i.type
       v.vehicleID, v.make, v.model, v.year
FROM Invoice i
JOIN RepairOrder ro ON i.repairOrderID = ro.repairOrderID
JOIN Vehicle v ON ro.vehicleID = v.vehicleID
WHERE i.invoiceID = 4;
```

	mileagein integer	mileageout integer	dateprinted date	datepaid date	type character varying	vehicleid integer	make character varyir	model character varying	year integer
1	129990	129996	2024-07-14	2024-07-15	Prepayment	4	Honda	Accord	2022

In conclusion, this project offered a comprehensive opportunity to design, develop, and implement a relational database system for a real-world case study. Throughout the process, I applied key database concepts, including data modeling, normalization, and relational design, while translating the conceptual model into a functional PostgreSQL schema. By developing SQL queries, ensuring data integrity, and refining the logical structure, I gained valuable hands-on experience in managing databases. This project also enhanced my ability to generate reports and interpret query results, reinforcing my understanding of database management systems and their practical applications.